

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

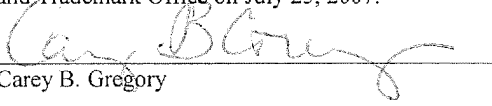
In re:	Brabson et al.	Confirmation No.: 3407
Serial No.:	10/007,581	Group Art Unit: 2135
Filed:	December 5, 2001	Examiner: Joseph T. Pan
For:	OFFLOAD PROCESSING FOR SECURITY SESSION ESTABLISHMENT AND CONTROL	

Date: July 25, 2007

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

CERTIFICATION OF TRANSMISSION

I hereby certify that this correspondence is being transmitted via the Office electronic filing system in accordance with § 1.6(a)(4) to the U.S. Patent and Trademark Office on July 25, 2007.


Carey B. Gregory

APPELLANTS' BRIEF ON APPEAL UNDER 37 C.F.R. §41.37

Sir:

This Appeal Brief is filed pursuant to the "Notice of Appeal to the Board of Patent Appeals and Interferences" filed May 25, 2007.

Real Party In Interest

The real party in interest is assignee International Business Machines, Inc., Armonk, New York.

Related Appeals and Interferences

Appellants are aware of no appeals or interferences that would be affected by the present appeal.

Status of Claims

Appellants appeal the final rejection of Claims 1-12, 14, 16-18, 20 and 22-39, which remain under consideration as of the filing date of this Brief. The claims at issue, as included in Appellants' Request for Reconsideration filed on April 20, 2007, are attached hereto as Appendix A.

Status of Amendments

The application was originally filed with Claims 1-35. An Amendment was filed on August 22, 2005 in response to an Office Action mailed on July 12, 2005. In the August 22, 2005 Amendment, Claims 1, 2, 30, and 33-35 were amended, and new Claims 36-39 were added. Claims 13, 15, 19 and 21 were cancelled. No further amendments have been made. Accordingly, Claims 1-12, 14, 16-18, 20 and 22-39 remain for consideration in the present appeal.

Summary of Claimed Subject Matter

Appellants appeal the final rejection of Independent Claims 1 and 33-35. Independent Claim 1 is directed to a method of performing security processing in a computing network comprising a local unit having an operating system kernel executing at least one application program (Application layer of Fig. 2F). The method includes receiving a first request at the operating system kernel from the application program to initiate a communication with a remote unit (Fig. 2F; Specification, page 19, line 8 to page 20, line 10). A second request is provided from the operating system kernel to a security offload component which performs security handshake processing (Arrow 240 of Fig. 2F; Specification, page 19, lines 16-20). The second request directs the security offload component to secure the communication with the remote unit (Specification, page 19, lines 16-18). A control function is provided in the operating system kernel for initiating operation of the security handshake processing by the security offload component (Fig. 2F; Specification, page 19, line 8 to page 20, line 3).

Thus, for example, security processing operations may be performed transparently by the operating system kernel on behalf of an application program that may have no awareness of security processing. See Specification, page 7, lines 17-18 and page 19, line 20 to page 20, line

3. Moreover, methods according to Claim 1 can provide systems in which clear text is available throughout the stack. See Specification, page 20, lines 4-10. Furthermore, control of the security processing can be maintained in the operating system kernel even though the security processing itself is offloaded to the security offload component. See Specification, page 21, line 15 to page 22, line 6.

Independent Claim 33 is directed to a method of performing security processing in a computing network including a local unit having an operating system kernel executing at least one application program. The method includes providing a security offload component which performs security session establishment and control processing (Component 230 of Fig. 2F), and providing a control function in the operating system kernel for initiating operation of the security session establishment and control processing by the security offload component (Kernel based SSL control of Fig. 2F). A request is received at the operating system kernel from the application program to initiate a communication with a remote unit (Fig. 2F, Specification, page 19, line 8 to page 20, line 10), and the security offload component is directed to secure the communication with the remote unit in response to the request (Arrow 240 of Fig. 2F; Specification, page 19, lines 16-20).

Independent Claim 34 is directed to a system for performing security processing in a computing network including a local unit having an operating system kernel executing at least one application program. The system includes means for performing security session establishment and control processing in a security offload component. Structure corresponding to the "means for performing security session establishment and control processing in a security offload component" is provided, inter alia, by the hardware accelerator 230 230 of Fig. 2F. The system further includes means for executing a control function in the operating system kernel, thereby initiating operation of the means for performing security session establishment and control processing by the security offload component, means for receiving a request at the operating system kernel from the application program to initiate a communication with a remote unit, and means for directing the security offload component to secure the communication with the remote unit in response to the request. Structure corresponding to the "means for executing a control function in the operating system kernel," the "means for receiving a request at the operating system kernel from the application program to initiate a communication with a remote

unit," and the "means for directing the security offload component to secure the communication with the remote unit in response to the request" is provided, inter alia, by the Kernel based SSL control layer of Fig. 2F, which can implemented, for example, in a server device. See Specification page 19, line 8 to page 20, line 10.

Independent Claim 35 is directed to a computer program product for performing security processing in a computing network including a local unit having an operating system kernel executing at least one application program. The computer program product is embodied on one or more computer-readable media, and comprises computer-readable program code configured to perform security session establishment and control processing in a security offload component (Component 230 of Fig. 2F).

The computer program product further comprises computer-readable program code configured to execute a control function in an operating system kernel (Kernel based SSL control of Fig. 2F), thereby initiating operation of the computer-readable program code configured to perform security session establishment and control processing by the security offload component.

The computer program product further comprises computer-readable program code configured to receive a request at the operating system kernel from the application program to initiate a communication with a remote unit (Fig. 2F, Specification, page 19, line 8 to page 20, line 10), and computer-readable program code configured to direct the security offload component to secure the communication with the remote unit in response to the request (Arrow 240 of Fig. 2F; Specification, page 19, lines 16-20).

Grounds of Rejection to be Reviewed on Appeal

Claims 1-12, 14, 16-18, 20 and 22-39 stand rejected under 35 USC § 103(a) as unpatentable over U.S. Patent No. 6,141,705 to Anand et al. ("Anand") in view of U.S. Publication No. 2003/0014623 to Freed et al. ("Freed"). See Final Office Action, page 2.

Argument

I. Introduction to 35 U.S.C. §103 Analysis

Claims 1-12, 14, 16-18, 20 and 22-39 stand rejected as obvious under 35 U.S.C. §103(a). A determination under §103 that an invention would have been obvious to someone of ordinary skill in the art is a conclusion of law based on fact. *Panduit Corp. v. Dennison Mfg. Co.* 810 F.2d 1593, 1 U.S.P.Q.2d 1593 (Fed. Cir. 1987), *cert. denied*, 107 S.Ct. 2187. After the involved facts are determined, the decision maker must then make the legal determination of whether the claimed invention as a whole would have been obvious to a person having ordinary skill in the art at the time the invention was unknown, and just before it was made. *Id.* at 1596. The United States Patent and Trademark Office (USPTO) has the initial burden under §103 to establish a *prima facie* case of obviousness. *In re Fine*, 837 F.2d 1071, 5 U.S.P.Q.2d 1596, 1598 (Fed. Cir. 1988).

To establish a *prima facie* case of obviousness, the prior art reference or references when combined must teach or suggest *all* the recitations of the claims M.P.E.P. §2143. Furthermore, as recently stated by the U.S. Supreme Court, *KSR International Co. v. Teleflex Inc., et al.*, 550 U.S. 1, 14 (2007), there must be some reason to combine the references in a way that produces the claimed invention, and a patent composed of several elements is not proved obvious merely by demonstrating that each of its elements was, independently, known in the prior art.

Appellants respectfully submit that the pending claims are patentable over the cited references for at least the reason that neither the cited references nor the combination thereof disclose or suggest each of the recitations of the claims. The patentability of the pending claims is discussed in detail hereinafter.

II. Independent Claims 1 and 33-35 are Patentable over Anand in view of Freed

A. Claim 1 Is Patentable over Anand in view of Freed

Claim 1 recites as follows (emphasis added):

1. A method of performing security processing in a computing network comprising a local unit having an operating system kernel executing at least one application program, comprising:

receiving a first request **at the operating system kernel** from the application program to initiate a communication with a remote unit;

providing a second request **from the operating system kernel** to a security offload component which performs security handshake processing, the second request

directing the security offload component to secure the communication with the remote unit; and

providing a control function **in the operating system kernel** for initiating operation of the security handshake processing by the security offload component.

Accordingly, Claim 1 is directed to a method of performing security processing in a computing network in which operation of security handshake processing by a security offload component is initiated by a control function in an operating system kernel. As explained in the present application, such operations may be performed transparently by the operating system kernel on behalf of an application program that may have no awareness of security processing. See Specification, page 7, lines 17-18 and page 19, line 20 to page 20, line 3.

The September 13, 2006 Non-Final Office Action (the "Non-Final Office Action") cited Figure 3 and the accompanying text of Anand at col. 10, ll. 27-47 as disclosing the claimed steps of receiving a first request at the operating system kernel from the application program to initiate a communication with a remote unit, providing a second request from the operating system kernel to a security offload component directing the security offload component to secure the communication with the remote unit, and providing a control function in the operating system kernel for initiating operation of the security handshake processing by the security offload component. Non-Final Office Action, pages 2-3.

However, in distinct contrast to the recitations of Claim 1, Anand is directed to a system by which security processing is performed by an offload component (e.g. a NIC) under the direction and control of an application program, not an operating system kernel. As explained in Anand, "[a]n **application executing on the computer system** first queries the processing, or task offload capabilities of the NIC, and then selectively enables those capabilities that may be subsequently needed by the application." Anand, Abstract. Furthermore, Anand states that "[o]nce an **application** has discerned the capabilities of a particular NIC, it will selectively utilize any of the enabled task offload capabilities of the NIC by appending packet extension data to the network data packet that is forwarded to the NIC." Anand, Abstract (emphasis added). Thus, Anand explicitly requires the application program to query the capabilities of an offload processor and then selectively enable those capabilities.

Accordingly, Figure 3 of Anand does not teach or suggest kernel-based offload security processing as recited in Claim 1. In fact, the system of Anand Figure 3 expressly **excludes** kernel-based offload security processing as recited in Claim 1. For example, the Non-Final Office Action cites element 128 of Anand Figure 3 as teaching the step of providing a second request from the operating system kernel to a security offload component directing the security offload component to secure the communication with the remote unit. However, element 128 of Anand Figure 3 is explicitly labeled a "Transport Protocol **Driver**," which a skilled person would necessarily understand to be different from an operating system kernel.

The Non-Final Office Action further cited element 100 of Anand Figure 3 as providing the claimed step of providing a control function in the operating system kernel for initiating operation of the security handshake processing by the security offload component. Non-Final Office Action, page 3. However, element 100 of Anand Figure 3 is simply the NIC, which in the system of Anand communicates with a Network **Driver** 116, not with an operating system kernel. In fact, Anand expressly differentiates the network drivers from the operating system. See Anand, col. 8, ll. 26-31 ("Each driver, typically implemented as a software component provided by the vendor of the corresponding NIC, is responsible for sending and receiving packets over its corresponding network connection and for managing the NIC on behalf of the operating system.").

In response to the Appellant's arguments, the Final Office Action of February 28, 2007 (the "Final Office Action") cited the following passage of Anand at col. 3, lines 9-23:

In a preferred embodiment of the invention, a software implemented method and protocol is provided that allows, for instance, the operating system (OS) to "query" the device drivers (often referred to as "MAC" drivers) of any hardware peripherals (such as a NIC) that are connected to the computer system. The various device drivers each respond by identifying their respective hardware peripheral's processing capabilities, referred to herein as "task offload capabilities." In the preferred embodiment, once the task offload capabilities of each particular peripheral have been identified, the OS can then enable selected peripherals to perform certain tasks that could potentially be used by the OS. The OS can thereafter request that a peripheral perform the previously enabled task, or tasks, in a dynamic, as-needed basis, depending on the then current processing needs of the computer system.

Final Office Action, page 9. Appellants submit that the foregoing passage of Anand, which is described by Anand as the "general inventive concept" (Anand, col. 1, line 24), does not mean

that in the system of Anand, an operating system kernel can initiate a request to a security offload component to secure a particular communication with a remote unit in response to receiving a request from an application program to initiate a communication with the remote unit, as recited in Claim 1. Unlike the method recited in Claim 1, the actual control of the particular security processing to be performed is directed by a transport protocol driver, as described, for example, at col 10, lines 27-63 of Anand. Nothing in the cited passage indicates that the operating system kernel can direct a peripheral to secure a particular communication with a particular remote unit, much less to initiate operation of the security handshake processing by the security offload component, as recited in Claim 1.

Instead, the description of Anand makes it abundantly clear that security processing is performed according to requests from a transport protocol driver which, as Appellants previously explained, is not part of the operating system kernel.

The Final Office Action asserted that Anand discloses that the transport protocol driver is implemented in the operating system kernel. Final Office Action, page 10. In support, the Final Office Action cited the following passage of Anand:

In a preferred embodiment of the present invention, in the Windows NT layered networking architecture, a transport protocol driver, or transport, is implemented with an appropriate program method so as to be capable of querying each of the device driver(s) associated with the corresponding NIC(s) connected to the computer.

Anand, col. 3, ll. 45-50. The conclusion of the Final Office Action, namely, that the transport protocol driver is implemented in the operating system kernel, is incorrect. The cited passage describes the "Windows NT layered networking architecture," not the Windows NT operating system. Moreover, the passage states that the transport protocol driver "is implemented with an appropriate program method," and does not indicate that the driver is somehow implemented in the operating system kernel.

In contrast, Anand distinguishes between the operating system and the transport protocol driver. For example, as stated by Anand "[m]ore particularly, NDIS describes the interface by which one or multiple NIC drivers (116-120) communicate with one or multiple underlying NICs (100-104), one or multiple overlying transport protocol drivers, or transports, (represented

at 128-134 in FIG. 2), and the operating system." (Emphasis added) Anand, col. 8, ll. 50-55. Thus, the transport protocol drivers are treated by Anand as distinct from the operating system. Anand goes on to say:

All interactions between NIC driver and protocol driver, NIC driver and operating system, and NIC driver and NIC are executed via calls to wrapper functions. Thus, instead of writing a transport-specific driver for Windows NT, network vendors provide the NDIS interface as the uppermost layer of a single network driver. Doing so allows any protocol driver to direct its network requests to the network card by calling this interface. Thus, a user can communicate over a TCP/IP network and a DLC (or an NWLINK, or DECnet, VINES, NetBEUI and so forth) network using one network card and a single network driver. (Emphasis added).

Anand, col. 8, l. 65 to col. 9, l. 8. Thus, the network driver, i.e. the transport protocol driver, is provided by the network vendor, and is clearly not part of the operating system kernel.

The Advisory Action of May 1, 2007 (the "Advisory Action") cited the following passage of Anand:

FIG. 6 illustrates one presently preferred set of program steps for implementing this ability to query the task offload capabilities of a peripheral, such as a NIC, and for then setting/enabling those tasks that may be needed. Preferably, the illustrated program steps are implemented as a software component or module that is integrated within a transport protocol driver. Moreover, if implemented in a Windows NT environment, many of the functional operations and inter-driver communications are preferably carried out via the NDIS interface. Of course, if implemented in a different operating system environment, or implemented in connection with non-network types of peripherals, then the illustrated program steps would have to be modified accordingly.

Anand, col. 13, lines 44-56. The Advisory Action asserted that, as used therein, the term "Windows NT environment" refers to the Windows NT operating system. See Advisory Action, Continuation Sheet. Appellants submit that interpretation of Anand expressed in the Final Office Action and the Advisory Action clearly ignores the term "environment," which is understood by a skilled person not to refer to the operating system itself, but to the environment created by the operating system in which application programs operate. That is, in the context of computer systems, the term "environment" refers, not to the operating system kernel, but to a computer interface from which various tasks can be performed.

Furthermore, the cited passage of Anand is consistent with the Appellants' interpretation of the term "Windows NT environment," as the cited passage discusses implementation of drivers that perform various tasks on a computer. Accordingly, a skilled person would not understand the term "Windows NT environment" used in the cited passage of Anand to refer to the Windows NT operating system kernel, but rather would understand the term to refer to a Windows NT computer interface from which various tasks can be performed.

This understanding of the term "Windows NT environment" is consistent with Appellants' understanding of the teaching of Anand, namely, that in Anand a transport protocol driver, as opposed to the operating system kernel, can issue requests for security offload processing. Anand's teaching therefore contrasts sharply with the recitations of Claim 1 that provide "receiving a first request at the operating system kernel from the application program to initiate a communication with a remote unit, providing a second request from the operating system kernel to a security offload component which performs security handshake processing, the second request directing the security offload component to secure the communication with the remote unit, and providing a control function in the operating system kernel for initiating operation of the security handshake processing by the security offload component" (emphasis added). Moreover, Anand does not suggest the benefits of controlling the operation of a security offload component using a control function in an operating system kernel rather than requiring such control functionality to be implemented in an application program.

Having established that the network driver of Anand is not part of the operating system, Anand teaches that the security functions of a NIC 100 may be invoked by the driver by appending an appropriate packet extension to a data packet. See Anand, col. 10, ll. 27-47. In particular, Anand states:

For instance, in FIG. 3 the application data 140 is passed down from the upper layers of the network model to an appropriate transport protocol driver, such as TCP/IP 128. The driver repackages the data into an appropriate data packet 142. Then, depending on whatever additional functions are to be performed on this particular data packet 142 a functional component is included that appends a predefined data structure, referred to as the packet extension, to the data packet. (Emphasis added).

Anand, col. 10, ll. 35-38.

Accordingly, Anand expressly teaches away from an operating system kernel directing operations of a security offload component as recited in Claim 1.

The Non-Final Office Action cited Freed as disclosing a method for secure communications between a client and a server including managing a communication negotiation between the client and the server. Non-Final Office Action, p. 3. Thus, Freed was not cited as providing the missing recitations noted above, and even if combined, Freed and Anand would not teach every element of Claim 1.

B. Claim 33 Is Patentable over Anand in view of Freed

Independent Claim 33 recites (emphasis added):

33. A method of performing security processing in a computing network including a local unit having an operating system kernel executing at least one application program, comprising:

providing a security offload component which performs security session establishment and control processing;

providing a control function in the operating system kernel for initiating operation of the security session establishment and control processing by the security offload component;

receiving a request at the operating system kernel from the application program to initiate a communication with a remote unit; and

directing the security offload component to secure the communication with the remote unit in response to the request.

For at least the reasons discussed above, Appellants respectfully submit that Anand does not teach or suggest providing a control function in an operating system kernel for initiating operation of security session establishment and control processing by a security offload component, receiving a request at the operating system kernel from the application program to initiate a communication with a remote unit, or directing the security offload component to secure the communication with the remote unit in response to the request, as recited in Claim 33. The Final Office Action cites Freed as disclosing a method for secure communications between a client and a server including managing a communication negotiation between the client and the

server. Final Office Action, page 3. Thus, Freed is not cited as providing the missing recitations noted above, and even if combined, Freed and Anand would not teach every element of Claim 33.

C. Claims 34-35 Are Patentable over Anand in view of Freed

Independent Claims 34-35 are system and computer program product claims containing recitations similar to those of Independent Claim 33, and are submitted to be patentable for at least similar reasons as Claim 33.

For at least the foregoing reasons, Appellants respectfully submit that independent Claims 1 and 33-35 are patentable over Anand in view of Freed, and that dependent Claims 2-12, 14, 16-18, 20, 22-32 and 36-39 are patentable at least by virtue of their depending from an allowable claim. Accordingly, Appellants respectfully request that the rejection of Claims 1-12, 14, 16-18, 20 and 22-39 be reversed, and that the application be passed to allowance.

III. Conclusion

In summary, Appellants respectfully submit that the cited references do not teach or suggest all of the recitations of the claims, either alone or in combination. Accordingly, Appellants respectfully request reversal of the rejection of Claims 1-12, 14, 16-18, 20 and 22-39 based on the cited references.

Respectfully submitted,

A handwritten signature in dark ink, appearing to read "D. Hall", with a stylized flourish at the end.

David C. Hall
Registration No. 38,904

Myers Bigel Sibley & Sajovec, P.A.
P. O. Box 37428
Raleigh, North Carolina 27627
Telephone: (919) 854-1400
Facsimile: (919) 854-1401
Customer No. 46589

APPENDIX A - Claim Listing

1. (Previously presented) A method of performing security processing in a computing network comprising a local unit having an operating system kernel executing at least one application program, comprising:

receiving a first request at the operating system kernel from the application program to initiate a communication with a remote unit;

providing a second request from the operating system kernel to a security offload component which performs security handshake processing, the second request directing the security offload component to secure the communication with the remote unit; and

providing a control function in the operating system kernel for initiating operation of the security handshake processing by the security offload component.

2. (Previously presented) The method according to Claim 1, further comprising executing the provided control function, thereby initiating operation of the security handshake processing.

3. (Original) The method according to Claim 1, wherein the operating system kernel maintains control over operation of the security handshake processing.

4. (Original) The method according to Claim 1, wherein the operating system kernel does not participate in operation of the security handshake processing.

5. (Original) The method according to Claim 1, wherein the control function further specifies information to be used by the security offload component during the security handshake processing.

6. (Original) The method according to Claim 5, wherein the specified information comprises one or more of: a connection identifier; a security role; one or more security versions supported; and cipher suites options.

7. (Original) The method according to Claim 1, wherein:
the operating system kernel does not participate in operation of the security handshake processing;
the control function further specifies information to be used by the security offload component during the security handshake processing; and
the specified information comprises one or more of: a connection identifier; a security role; one or more security versions supported; cipher suites options; and security certificate key ring information.

8. (Original) The method according to Claim 7, wherein the specified information further comprises segment size and sequence number information to be used when transmitting messages of the security handshake processing.

9. (Original) The method according to Claim 7, further comprising the step of sending a completion response from the security offload component to the operating system kernel upon completion of the security handshake processing, wherein the completion response conveys information for use by the operating system kernel in carrying out secure communications on a secure session which results from the security handshake processing.

10. (Original) The method according to Claim 9, wherein the conveyed information comprises one or more of: an identifier of the secure session; one or more session keys; a current sequence number for messages of the secure session; a cipher suite to be used for the secure session; a protocol version to be used for the secure session; and a digital certificate or other security credentials.

11. (Original) The method according to Claim 1, wherein the operating system kernel maintains control over operation of the security handshake processing, and wherein the operating system kernel provides one or more message segments to the security offload component for use by the security offload component in completing steps of the security handshake processing.

12. (Original) The method according to Claim 11, wherein a selected one of the one or more message segments directs the security offload component in a client device to perform random number generation when creating an initial handshake message to transmit to a server device.

13. (Cancelled).

14. (Original) The method according to Claim 11, wherein a selected one of the one or more message segments directs the security offload component in a server device to perform random number generation when creating an initial handshake response message to transmit to a client device.

15. (Cancelled).

16. (Original) The method according to Claim 11, wherein a selected one of the one or more message segments directs the security offload component in a server device to decode a client security certificate which has been transmitted from a client device.

17. (Original) The method according to Claim 11, wherein a selected one of the one or more message segments directs the security offload component in a client device to decode a server security certificate which has been transmitted from a server device.

18. (Original) The method according to Claim 11, wherein a selected one of the one or more message segments directs the security offload component in a client device to generate and encrypt a pre-master security secret to be transmitted to a server device.

19. (Cancelled).

20. (Original) The method according to Claim 11, wherein a selected one of the one or more message segments directs the security offload component in a server device to decrypt a pre-master security secret transmitted from a client device.

21. (Cancelled).

22. (Original) The method according to Claim 11, wherein a selected one of the one or more message segments directs the security offload component in a client device to compute one or more master security secrets and one or more session cryptography keys to be transmitted to a server device.

23. (Original) The method according to Claim 11, wherein a selected one of the one or more message segments directs the security offload component in a server device to compute one or more master security secrets and one or more session cryptography keys to be transmitted to a client device.

24. (Original) The method according to Claim 11, wherein a selected one of the one or more message segments directs the security offload component in a client device to digitally sign a message to be transmitted to a server device.

25. (Original) The method according to Claim 11, wherein a selected one of the one or more message segments directs the security offload component in a server device to validate a digital signature of a message received from a client device.

26. (Original) The method according to Claim 11, wherein a selected one of the one or more message segments directs the security offload component in a client device to compute a message authentication code ("MAC") of the security handshake, wherein the computed MAC is to be transmitted to a server device.

27. (Original) The method according to Claim 11, wherein a selected one of the one or more message segments directs the security offload component in a server device to compute a message authentication code ("MAC") of the security handshake, wherein the computed MAC is to be transmitted to a client device.

28. (Original) The method according to Claim 11, wherein a selected one of the one or more message segments directs the security offload component in a client device to validate a message authentication code ("MAC") of the security handshake, wherein the MAC was transmitted from a server device.

29. (Original) The method according to Claim 11, wherein a selected one of the one or more message segments directs the security offload component in a server device to validate a message authentication code ("MAC") of the security handshake, wherein the MAC was transmitted from a client device.

30. (Previously presented) The method according to Claim 11, further comprising sending a completion response from the security offload component to the operating system kernel upon completion of the security handshake processing, wherein the completion response conveys information for use by the operating system kernel in carrying out secure communications on a secure session which results from the security handshake processing.

31. (Original) The method according to Claim 30, wherein the conveyed information comprises one or more of: an identifier of the secure session; one or more session keys; a current sequence number for messages of the secure session; a cipher suite to be used for the secure session; a protocol version to be used for the secure session; and a digital certificate or other security credentials.

32. (Original) The method according to Claim 31, wherein the conveyed information further comprises a current transmission control sequence number for transmitting messages of the secure session.

33. (Previously presented) A method of performing security processing in a computing network including a local unit having an operating system kernel executing at least one application program, comprising:

providing a security offload component which performs security session establishment and control processing;

providing a control function in the operating system kernel for initiating operation of the security session establishment and control processing by the security offload component;

receiving a request at the operating system kernel from the application program to initiate a communication with a remote unit; and

directing the security offload component to secure the communication with the remote unit in response to the request.

34. (Previously presented) A system for performing security processing in a computing network including a local unit having an operating system kernel executing at least one application program, comprising:

means for performing security session establishment and control processing in a security offload component;

means for executing a control function in the operating system kernel, thereby initiating operation of the means for performing security session establishment and control processing by the security offload component;

means for receiving a request at the operating system kernel from the application program to initiate a communication with a remote unit; and

means for directing the security offload component to secure the communication with the remote unit in response to the request.

35. (Previously presented) A computer program product for performing security processing in a computing network including a local unit having an operating system kernel executing at least one application program, the computer program product embodied on one or more computer-readable media and comprising:

computer-readable program code configured to perform security session establishment and control processing in a security offload component;

computer-readable program code configured to execute a control function in an operating system kernel, thereby initiating operation of the computer-readable program code configured to perform security session establishment and control processing by the security offload component; and

computer-readable program code configured to receive a request at the operating system kernel from the application program to initiate a communication with a remote unit; and

computer-readable program code configured to direct the security offload component to secure the communication with the remote unit in response to the request.

36. (Previously presented) The method according to claim 1, further comprising:
preparing a data packet including data to be communicated to the remote unit;
reserving space in the data packet for security protocol information; and
passing the data packet including the reserved space to the security offload component.

37. (Previously presented) The method according to claim 36, further comprising:
passing control information from the operating system kernel to the security offload component, wherein the control information is passed to the security offload component in the space reserved in the data packet for security protocol information.

38. (Previously presented) The method according to claim 36, further comprising:
passing control information from the operating system kernel to the security offload component, wherein the control information is passed to the security offload component separately from the data packet.

39. (Previously presented) The method according to claim 36, further comprising:
receiving the data packet with the reserved space at the security offload component;
encrypting the data in the data packet;

inserting security protocol information in the reserved space; and
transmitting the resulting data packet to the remote unit.

In re: Brabson et al.
Serial No.: 10/007,581
Filed: December 5, 2001
Page 21

APPENDIX B – EVIDENCE APPENDIX

None.

In re: Brabson et al.
Serial No.: 10/007,581
Filed: December 5, 2001
Page 22

APPENDIX C – RELATED PROCEEDINGS APPENDIX

None.